0670-5782US/01-7-16
File: 0670-5782usf/SUE

TITLE

MODULAR MULTIPLIER AND AN ENCRYPTION/DECRYPTION PROCESSOR USING THE MODULAR MULTIPLIER

BACKGROUND OF THE INVENTION

Field of the Invention:

This invention relates to a modular multiplier operation structure, particularly to a modular multiplier realized by the high-radix Montgomery operation algorithm.

Description of the Related Art:

Due to the requirements of data transfer in networking and digitization, the cryptography for the data security mechanism has spurred efforts to the design. The basic principle of the cryptography is that a plaintext is converted into a ciphertext through a encryption and a encryption key chosen by a user. When a receiver receives the ciphertext, a decryption with respect to the encryption and a respective decryption key of the encryption key can recover the plaintext. Because the data in transfer or storage is in the ciphertext, the data security is achieved since an adversary has no the decryption key to get the transfer data.

The security of a cryptosystems is built on the basis of the potential of extracting the keys. The security of the cryptosystem is indicated by the potential of extracting the keys from the existing data. Current cryptosystem is divided into two types, private key cryptosystem and public

key cryptosystem. In private key cryptosystem, encryption and decryption keys are the same, for example, the widely used system is the DES system. The same encryption and decryption keys mean that the keys must be stored in an absolutely secure transmission path to ensure the transfer security. This is the main drawback in private key crytosystem. There is no such a problem in public key cryptosystem. In public key cryptosystem, encryption and decryption keys are different. In a pair of encryption and decryption keys, encryption key is a public key. When the plaintext is encrypted by encryption key into the ciphertext, only the respective decryption key of encryption key can recover it. Also, such a system, e.g. Rivest, Shamir, Adleman (RSA), must offer the guaranty that the respective decryption can not or hardly be extracted without telling it. Accordingly, public key cryptosystem becomes increasingly and leads the world trend in the cryptosystem because besides it has not the key transfer and management problem, the descryption key in public key cryptosystem offers the function of certifying a digital signature.

RSA cryptosystem uses the modular exponentiation operation to generate the encryption/decryption function. The encryption/decryption is expressed as follows:

$$C=M^{E}(\text{mod } N) \qquad (1)$$

$$M=C^{D}(\text{mod } N) \qquad (2)$$

where N=PQ and ED 1 mod(P-1)(Q-1), M is plaintext; C is ciphertext; E is encryption key; and D is decryption key.

N is the product of two prime numbers P and Q. Equation (1) represents the encryption action. The modular multiplication operation (E, N) is used to convert the plaintext M into the ciphertext C. Equation (2) represents

the decryption action. The modular multiplication operation (D, N) is used to recover the plaintext M from the ciphertext C. In RSA cryptosystem, the modular exponentiation operation is complex and takes much time in computation. Hence, the modular multiplier is commonly used to realize the modular exponentiation operation, especially to the utilization of Montgomery algorithm. For example, the Montgomery algorithm is used in the basic operation of AB(mod N) as the following algorithm 1:

<Algorithm 1>

$R_0=0$;

For i=0 to n-1 do

$q_i=R_i+a_iB \pmod 2$ (3)

$R_{i+1}=(R_i+a_iB+q_iN)/2$ (4)

end

where $A=a_{n-1}2^{n-1}+a_{n-2}2^{n-2}+\ldots\ldots+a_0$; $B=b_{n-1}2^{n-1}+b_{n-2}2^{n-2}+\ldots\ldots+b_0$; and $a_i,b_i,q_i\in\{0, 1\}$.

The foregoing algorithm performs a n-time loop with an n-bit adder and a 1xn multiplier. The performed result for every loop is respectively multiplied by 2^0, 2^1, 2^2,……,2^{n-1} and then summed total. The final total summed is expressed as follows:

$2^nR_n = AB \pmod N$ (5)

According to equation (5), R_n is expressed as follows:

$R_n = 2^{-n}AB \pmod N$ (6)

Therefore, the modular exponentiation operation of equation (1) or (2) is performed by Montgomery algorithm according to the following pre-operation, exponentiation operation, and post-operation:

$MGM(M, 2^{2n}) = 2^nM \pmod N$ (7)

$MGM(2^nM^a, 2^nM^b) = 2^nM^{a+b} \pmod N$ (8)

$$MGM(2^nM^E, 1) = M^E \pmod N \qquad (9)$$

where $MGM(\bullet,\bullet)$ represents the operand R_n executed by Montgomery algorithm, i.e., the result from equation (6) $R_n = 2^{-n}AB \pmod N$.

Because the need of performing n-time loop in algorithm 1 takes time in the computation, the chip area in the high radix(2^k) Montgomery algorithm is adopted to efficiently increase the operation speed. The high radix Montgomery algorithm reduces the modular multiplication from one to n/k by dividing the operand A into [n/k] groups, each group having k bits, when decoding or encoding data, thereby achieving the purpose of increasing the speed. The algorithm is expressed as follows:

<Algorithm 2>

$R_0=0$;

For i=0 to [n/k]-1 do

$$q_i=(R_i+a_iB)*N_1 \pmod{2^k} \qquad (10)$$

$$R_{i+1}=(R_i+a_iB+q_iN)/2^k \qquad (11)$$

end

where N_1 is satisfied with $N*N_1 \equiv -1 \pmod{2^k}$, $A=a_{\lceil n/k\rceil-1}(2^k)^{\lceil n/k\rceil-1}+a_{\lceil n/k\rceil-2}(2^k)^{\lceil n/k\rceil-2}+\cdots\cdots+a_0$; and a_i, $q_i \in \{0, 1, 2, \cdots\cdots, 2^{k-1}\}$, for k >0.

Although the loop in algorithm 2 is reduced, a further reduction for the loop is subjected to algorithm 3, which shifts the operand B by k bits and changes the parameter N into N_2 in order to eliminate the multiplication and addition operations in equation (10). The expression is:

<Algorithm 3>

$R_0=0$;

For i=0 to $\lceil n/k \rceil$ do

$$q_i=R_i \pmod{2^k} \qquad (12)$$

$$R_{i+1}=(R_i+q_i*N_2)/2^k+a_iB \qquad (13)$$

end

where $N_2 = mN \equiv -1 \pmod{2^k}$.

Likewise, the result for every loop is respectively multiplied by 2^0, 2^1, 2^2,.....,2^{n-1} and then summed total. The final total is expressed as follows:

$$2^{n+k}R_{\lceil n/k \rceil+1} \quad A*2^k*B+Q*N_2 \qquad (14)$$

Accordingly, the relationship derived from equation (5) is satisfied as a result of $R_{(n/k)+1}$ and that is:

$$2^nR_{\lceil n/k \rceil+1} \quad AB \pmod{N} \qquad (15)$$

The best advantage in algorithm 3 is the same operation structure as mentioned above, i.e., only a multiplication and addition is executed for the operand R_{i+1} in equation (13).

Assume that $X= R_i + q_i*N_2$ (16)

Then equation (13) is modified as the following equation:

$$R_{i+1}=X/2^k+a_i*B \qquad (17)$$

If $Y=X/2^k$, equation (17) is changed as the following equation:

$$R_{i+1}=Y+a_i*B \qquad (18)$$

Equations (17) and (18) are respectively executed a multiplication and addition operations and the corresponding operands have the same bit number. Therefore, a same data path is used in the computation operation at different time points, thereby saving the area required for a chip.

However, Montgomery algorithm 3 also has the complex computation problem when the required area for the multiplication is broad. In equations (16) and (18), a kxn multiplier is used. If the values n and k are large, for example, k=32 and n=1024, the chip area therefore becomes very broad. For a chip with the strict request of small

size, e.g. a Smart Card, this will influence on its operation and application. As to this point, the invention provides a solution by improving the high radix Montgomery algorithm to reduce the chip area and have the high-speed operation.

SUMMARY OF THE INVENTION

Accordingly, the object of the invention is to provide a modular multiplier and an encryption/decryption processor using the modular multiplier, capable of reducing the chip area and achieving the purpose of high-speed operation.

To realize the above and other objects, the invention provides a modular multiplier, capable of processing a first operand and a second operand in relation to a modulus for performing the modular multiplication operation. The performed operation includes an instruction, which has an internal multiplication and addition operation with inner recursion and an external multiplication and addition operation. The modular multiplier includes a first buffer device for storing the first operand, the first operand is divided into a first plurality of sub-operands with fixed length; a second buffer device for storing the second operand, the second operand is divided into a second plurality of sub-operands with fixed length; a third buffer device for storing the parameter of the modular multiplication operation; a multiplexer device, coupled to the first, the second, and the third buffer devices, for choosing a first multiplication operand and a second multiplication operand from the first sub-operand, the second sub-operand, and the parameter in order according to the required internal and external multiplication/addition

operations; a multiplication device, coupled to the multiplexer device, for multiplying the first multiplication operand by the second multiplication operand to obtain a product; and an addition device, coupled to the multiplication device, for outputting an intermediate result according to the product during the internal multiplication and addition operation and outputting the result of the modular multiplication operation according to the product and the intermediate result during the external multiplication and addition operation.

The modular multiplier can be an encryption or decryption processor, for example, RSA cryption processor. The encryption or decryption processor performs the modular exponentiation operation in the encryption/decryption function according to the encryption/decryption key, thereby realizing the modular multiplier. The encryption/decryption processor can be applied to, such as, a Smart Card, especially to a modular multiplier having the needs of requiring a small chip area and higher operating speed.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram illustrating a modular multiplier of an embodiment of the invention;

Fig. 2 is a schematic diagram illustrating an adder to be operated in the first sub-loop according to the embodiment of the invention;

Fig. 3 is a schematic diagram illustrating an adder to be operated in the second sub-loop according to the embodiment of the invention;

Fig. 4 is a block diagram illustrating an RSA encryption/decryption processor realized by the modular multiplier of Fig. 1;

Fig. 5 is a schematic diagram illustrating the application of Fig. 4 in a Smart Card according to the embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

This invention provides a solution for reducing the chip area in the prior art. That is, in the prior art, algorithm 3 needs very broad chip area to implement a kxn multiplier. The following embodiment describes the inventive algorithm first and the modular multiplier structure in relation to the algorithm later.

In order to reduce the required chip area, the n-bit portion (i.e. the operand N_2 in equation (16) and the operand B in equation (18)) in algorithm 3 is grouped to $\lceil n/k \rceil$ groups, each group having k bits. That is,

<Algorithm 4>

$R_0=0$;

For i=0 to $\lceil n/k \rceil$ do

$$q_i=R_i \pmod{2^k} \qquad (19)$$

For j=0 to $\lceil n/k \rceil$-1 do

$$(R_{i+1})_j=((R_i)_j+q_i*(N_2)_j)/2^k+a_iB_j \qquad (20)$$

end

end

where $q_i*(N_2)_j$ and a_iB_j respectively are the kxk multiplication operation.

In algorithm 4, although the loop j needs extra carry and accumulation operations, the chip area is reduced obviously from kxn to kxk.

The algorithm 4 is further embodied in following algorithm 5:

<Algorithm 5>

$R_0=0$;

For i=0 to $\lceil n/k \rceil$ do

$q_i=R_i \bmod 2^k$ (21)

$W=q_i*(N_2)_0$ (22)

$C_{-1}=(R_i)_0 + W[(k-1):0]$ (23)

$V=0$ (24)

For j=0 to $\lceil n/k \rceil$-1 do

$Z=W$ (25)

$W=q_i*(N_2)_{j+1}$ (26)

$U=a_i*B_j$ (27)

$\{C_j,(R_{i+1})_j\}=(R_i)_{j+1}+W[(k-1):0]+Z[(2k-1):k]+U[(k-1):0]+V[(2k-1):k]+C_{j-1}$ (28)

$V=U$ (29)

end

end

where W, Z, U, V are temporary buffers, C_{-1}, C_j are carry bits, and $\{C_j,(R_{i+1})_j\}$ is the total of k-bit addition. More, $(R_i)_0 + W[(k-1):0]$ can become zero(i.e. $C_{-1}=0$) if choosing appropriate q_i, N_2.

In algorithm 5, two kxk multipliers are used to respectively calculate the operand W in equation (26) and the operand U in equation (27). In fact, algorithm 5 can further uses two sub-loop operations in loop j as following equation 6.

<Algorithm 6>

$R_0=0$;

For i=0 to $\lceil n/k \rceil$ do

$q_i=R_i \pmod{2^k}$ (30)

For j=0 to $\lceil n/k \rceil$-1 do

$Y_j=((R_i)_j + q_i*(N_2)_j)/2^k$ (31)

end

For j=0 to $\lceil n/k \rceil$-1 do

$(R_{i+1})_j=Y_j+a_i*B_j$ (32)

end

end

Likewise, algorithm 6 is further embodied in following algorithm 7:

<Algorithm 7>

$R_0=0$;

For i=0 to $\lceil n/k \rceil$ do

$q_i=R_i \bmod 2^k$ (33)

$W=q_i*(N_2)_0$ (34)

$C_{-1}=(R_i)_0 + W[(k-1):0]$ (35)

For j=0 to $\lceil n/k \rceil$-1 do

$Z=W$ (36)

$W=q_i*(N_2)_{j+1}$ (37)

$\{C_j, Y_j\}=(R_i)_{j+1}+W[(k-1):0]+Z[(2k-1):k]+C_{j-1}$ (38)

$C_{-1}=0$ (39)

$Z=0$ (40)

end

For j=0 to $\lceil n/k \rceil$-1 do

$W=a_i*B_j$ (41)

$\{C_j, (R_{i+1})_j\}=Y_j+W[(k-1):0]+Z[(2k-1):k]+C_{j-1}$ (42)

Z=W (43)

end

end

In algorithm 6 and 7, the loop j in algorithm 5 is divided into two sub-loops. This manner can reduce the requirement of two kxk multipliers to only one kxk multiplier, thereby shrinking the required chip area. Besides, the performance is even faster. For example, when n=1024, k=32, and a clock requirement to a 32x32 multiplication is assumed, executing the first sub-loop j in equation (31) needs (1024/32)=32 clocks and the same clocks as performing the second sub-loop j in equation (32). The entire multiplication operation (i.e. loop i) takes (1024/32+1)x(32+32)=2112 clocks. If the H-Algorithm is used in the 1024-bit RSA encode or decode modular exponentiation operation, the entire circuit takes about 2x2112x1024 clocks(about 4M clocks), i.e., $4n^2(n+1)/k^2$ in terms of parameters n and k. Thus, the purposes of smaller chip area and faster operation are achieved at the same time.

Fig. 1 is a block diagram illustrating a modular multiplier of equation 6 or 7. The modular multiplier structure in Fig. 1 is implemented according to algorithm 7, including buffers 101, 102, 103, 104, 105; multiplexers 201, 202; multiplier 203; control unit 204; flip/flops 301, 302, 303, 305, 306; and adder 304. Each element is described as follows.

Buffer 101 is used to store Montgomery algorithm's result $(R_{i+1})_j$ or the intermediate operand Y_j in the first sub-loop. Buffers 102-105 are used to respectively store the operands A, N_2, B, q_i of the two multiplication equations (equations (37) and (41)) in algorithm 7, wherein

operands A, N_2, B are a constant, a_i is a portion of bits of the operand A in i^{th} loop, $(N_2)_j$ and B_j are a portion of bits of operands N_2 and B in j^{th} loop. According to equation (33), q_i stored in buffer 105 is the remainder from $R_i/2^k$, that is, from bit (k-1) to bit 0 in R_i. Hence, the lower k bits of R_i stored in buffer 101 are extracted to have the operand q_i in buffer 105.

Multiplexers 201 and 202 are used to switch the required operands in the multiplication operation of different loop. For example, a multiplication operation is required for q_i and $(N_2)_j$ in equation (37) of the first sub-loop, while a multiplication operation is required for a_i and B_j in equation (41) of the second sub-loop. Multiplexers 201 and 202 are switched by the control signal CTRL of the control unit 204. A multiplication operation is performed by the kxk multiplier 203 with the outputs of 201 and 202 to create the product stored in buffer W with the length 2k.

Flip/flops 301-303 are used to store the result from the multiplier and output the result to the adder 304 to execute the addition operation in equations (38) and (42). Buffer W with the length 2k is divided into two k-length data, wherein the data in low bits W[(k-1):0] is outputted to flip/flop 302, the data in high bits W[(2k-1):k] is outputted to flip/flop 301. Flip/flop 303 stores the high bits Z[(2k-1):k] of the previous multiplication result. Flip/flop 305 stores the carry bit C_{j-1} of the previous addition result. Adder 304 performs the addition operation in equation (38) of the first sub-loop or in equation (42) of the second sub-loop. The difference between equations (38) and (42) for the addition operation is the operand, using $(R_i)_{j+1}$ or Y_j. When performing the first loop,

flip/flop 306 stores the operand Y_j while when performing the second loop, flip/flop 306 stores the operand $(R_i)_{j+1}$, and the two operands Y_j and $(R_i)_{j+1}$ are stored in buffer 101 temporarily.

The operation of the modular multiplier shown in Fig. 1 is described in detail as follows.

According to algorithm 7, the first instruction for every i loop begins with the calculation of the remainder of $R_i/2^k$, that is, taking lower k bits of the operand R_i in buffer 101 into buffer 105.

The operation starts the first sub-loop, which calculates Y_j with the parameters q_i, $(N_2)_j$, and $(R_i)_j$. First, in the 1st sub-loop, the parameter q_i in the ith loop is unchanged and comes from buffer 105 for the calculation. Buffer 103 outputs the corresponding $(N_2)_j$ depending on the value j. The higher k bits W[(2k-1):k] and lower k bits W[(k-1):0] of the product for every multiplication operation in the multiplier 203 are inputted to flip/flops 301 and 302, respectively. Inputting the higher k bits to flip/flop 301 is performed by a clock delay. Therefore, the performed result is counted into Y_{j+1} for the addition calculation. The value Y_j is calculated by the adder 304 to add together with the lower k bits W[(k-1):0], the higher k bits Z[(2k-1):k](stored in flip/flop 303) of previous product, $(R_i)_{j+1}$ (stored in buffer 101), and the overflow bit C_{j-1} of previous addition operation(stored in flip/flop 305). The calculated result from the adder 304 is stored in buffer 101 at next clock.

Fig. 2 is a schematic diagram illustrating an adder to be operated in the first sub-loop according to the embodiment of the invention. Assume that k=32 and n=1024,

the first column representing the calculation of equation (35). When j=0, the adder 304 adds up $R_i[63:32]$, $(q_i(N_2)_1)[31:0]$, $(q_i(N_2)_0)[63:32]$, and the carry bit C_{j-1} and gets Y[31:0]. When j=1, the adder 304 adds up $R_i[95:64]$, $(q_i(N_2)_2)[31:0]$, $(q_i(N_2)_1)[63:32]$, and the carry bit C_0 and gets Y[63:32]. The remaining operations for j=2 to 31 are all similar. That is, when j=31, Y[1023:992] is found, and Y[1023:0] is completed.

Thus, the second sub-loop sequentially starts at the calculation of $(R_{i+1})_j$ with the parameters a_i, B_j, Y_j. Likewise, the parameter a_i in the i^{th} loop is unchanged and comes from buffer 102 for the calculation. Buffer 104 outputs the corresponding B_j depending on the value j. The higher k bits W[(2k-1):k] and lower k bits W[(k-1):0] of the product for every multiplication operation in the multiplier 203 are inputted to flip/flops 301 and 302, respectively. Inputting the higher k bits to flip/flop 301 is performed by a clock delay. Therefore, the performed result is counted into $(R_{i+1})_{j+1}$ for the addition calculation. The value $(R_{i+1})_j$ is calculated by the adder 304 to add up the lower k bits W[(k-1):0], the higher k bits Z[(2k-1):k](stored in flip/flop 303) of previous product, Y_j(stored in buffer 101), and the carry bit C_{j-1} of previous addition operation(stored in flip/flop 305). The calculated result from the adder 304 is stored in buffer 101 at next clock.

Fig. 3 is a schematic diagram illustrating an adder to be operated in the second sub-loop according to the embodiment of the invention with reference to Fig. 2. When j=0, the adder 304 adds up Y[31:0], $(a_iB_1)[31:0]$ and $(a_iB_0)[63:32]$, and gets $R_{i+1}[63:32]$. The remaining

operations for j=1 to 31 are all similar. That is, when j=31, $R_{i+1}[1023:992]$ is found, and $R_{i+1}[1023:0]$ is completed.

Thus, repeated the calculation of R_i for every i and the final result of the Montgomery algorithm is found, which is the modular multiplication of $2^{-n}AB \pmod N$. It is noted that the intermediate content of corresponding flip/flops between the first and second sub-loops is clear in order to use the same data path to calculate different equations. The control unit 204 is used to control the entire operation by a control signal CTRL. The required calculation for the final result of equation 6 or 7 is performed by orderly shifting different multiplication operands into the multiplier.

The advantage of the invention is that the inventive modular multiplier can save the chip area and quickly perform the operation concurrently. Fig. 4 is a block diagram illustrating an RSA encryption/decryption processor realized by the modular multiplier of Fig. 1. As shown in Fig. 4, the RSA encryption/decryption processor includes an encryption/decryption core 12 and a modular multiplier core 14. The modular multiplier core 14 can be realized by, for example, the structure of Fig. 1. The modular multiplication result is calculated with the operands A and B. The encryption/decryption core 12 performs the required modular exponentiation operation to encrypt a plaintext to a ciphertext or decrypt the ciphertext to the plaintext using the steps of pre-operation in equation (7), exponentiation operation in equation (8) and post-operation in equation (9).

Fig. 5 is a schematic diagram illustrating the encryption/decryption structure applied to a Smart Card

according to the embodiment of the invention. Due to the limits to Smart Card's standard and its facility in carry, the strict chip area is a must. As shown in Fig. 5, the Smart Card 20 exchanges the external data through a communication interface 22. Before the data transfer, the data is encrypted by the encryption/decryption processor 24 through the internal memory 26 of the Smart Card 20 to ensure the data security. Because the need of finishing the required calculation as soon as possible by using the encryption/decryption processor 24 with a smaller area in a chip, the multiplier structure of the invention is the best choice to reach the goal.

Although the present invention has been described in its preferred embodiment, it is not intended to limit the invention to the precise embodiment disclosed herein. Those who are skilled in this technology can still make various alterations and modifications without departing from the scope and spirit of this invention. Therefore, the scope of the present invention shall be defined and protected by the following claims and their equivalents.